**SHEET- 14**

**Greedy Algorithm Approach**

1. There are N Mice and N holes are placed in a straight line. Each hole can accommodate only 1 mouse. A mouse can stay at his position, move one step right from x to x + 1, or move one step left from x to x -1. Any of these moves consumes 1 minute. Assign mice to holes so that the time when the last mouse gets inside a hole is minimized.

**Examples:**Input : positions of mice are:

4 -4 2

positions of holes are:

4 0 5

Output : 4

Assign mouse at position x = 4 to hole at position x = 4 : Time taken is 0 minutes

Assign mouse at position x=-4 to hole at position x = 0 : Time taken is 4 minutes

Assign mouse at position x=2 to hole at position x = 5 : Time taken is 3 minutes

After 4 minutes all of the mice are in the holes. Since, there is no combination possible where the last mouse's time is less than 4, answer = 4.

Input : positions of mice are:

-10, -79, -79, 67, 93, -85, -28, -94

positions of holes are:

-2, 9, 69, 25, -31, 23, 50, 78

Output : 102

1. Given an array arr[], find the lexicographically largest array that can be obtained by performing at-most k consecutive swaps.
2. Given a list of n integers containing numbers 1-n in a shuffled way and a integer K. N people are standing in a queue to play badminton. At first, the first two players in the queue play a game. Then the loser goes to the end of the queue, and the one who wins plays with the next person from the line, and so on. They play until someone wins k games consecutively. This player becomes the winner.

**Examples :**

Input: arr[] = {2, 1, 3, 4, 5}

k = 2

Output: 5

Explanation:

2 plays with 1, 1 goes to end of queue.

2 plays with 3, 3 wins, 2 goes to end of queue.

3 plays with 4, so 3 goes to the end of the queue.

5 plays with everyone and wins as it is the

largest of all elements.

Input: arr[] = {3, 1, 2}

k = 2

Output: 3

Explanation :

3 plays with 1. 3 wins. 1 goes to the end of the line.

3 plays with 2. 3 wins. 3 wins twice in a row.

1. Any number is called beautiful if it consists of 2N digits and the sum of the first N digits is equal to the sum of the last N digits. Your task is to find the count of beautiful numbers in the interval from L to R (including L and R). Beautiful numbers do not have leading zeroes.

**Input format**

* The first line contains an integer T denoting the number of test cases.
* The first line of each test case contains two space-separated integers L and R denoting the range interval [L,R].

**Output format**

For each test case, print the count of beautiful numbers in a new line.

**Constraints**

1≤T≤2000001≤L≤R≤1000000000

**Sample Input**

1

1 100

**Sample Output**

9

**Explanation**

There are only 9 beautiful numbers in the first 100 integers. 11,22,33,44,55,66,77,88 and 99 are the beautiful numbers in the range [1,100].

1. You are given a rectangle of height H and width W. You must divide this rectangle exactly into three pieces such that each piece is a rectangle of integral height and width. You are required to minimize Areamax−Areamin where Areamax is the area of the largest rectangle and Areamin is the area of the smallest rectangle, among all three rectangle pieces.

**Input format**

* The first line contains an integer T denoting the number of test cases.
* The first line of each test case contains two space-separated integers H and W denoting the height and width of the rectangle.

**Output format**

For each test case, print the minimum possible value of Areamax−Areamin in a new line.

**Constraints**

1≤T≤1000

2≤H,W≤200000

* It is guaranteed that the sum of H over T test cases does not exceed 1e6.
* It is guaranteed that the sum of W over T test cases does not exceed 1e6.

**Sample Input**

2

3 4

2 2

**Sample Output**

0

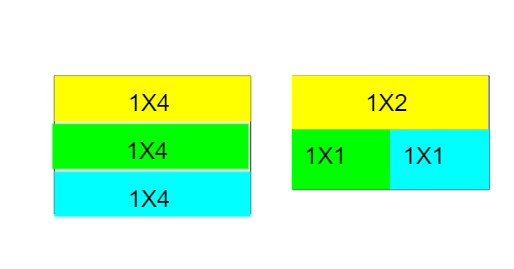
1

**Explanation**

For both the test cases, the division is shown below:

For the first testcase Areamax−Areamin = 4−4=0.

For the first testcase Areamax−Areamin = 2−1=1.



1. Given an array of size n that has the following specifications:
2. Each element in the array contains either a policeman or a thief.
3. Each policeman can catch only one thief.
4. A policeman cannot catch a thief who is more than K units away from the policeman.

We need to find the maximum number of thieves that can be caught.  
  
Examples:

**Input** : arr[] = {'P', 'T', 'T', 'P', 'T'},

k = 1.

**Output** : 2.

Here maximum 2 thieves can be caught, first

policeman catches first thief and second police-

man can catch either second or third thief.

**Input** : arr[] = {'T', 'T', 'P', 'P', 'T', 'P'},

k = 2.

**Output** : 3.

**Inpu**t : arr[] = {'P', 'T', 'P', 'T', 'T', 'P'},

k = 3.

**Outpu**t : 3.

1. You are given a string of 2N characters consisting of N ‘[‘ brackets and N ‘]’ brackets. A string is considered balanced if it can be represented in the for S2[S1] where S1 and S2 are balanced strings. We can make an unbalanced string balanced by swapping adjacent characters. Calculate the minimum number of swaps necessary to make a string balanced.

**Examples:**

Input : []][][

Output : 2

First swap: Position 3 and 4

[][]][

Second swap: Position 5 and 6

[][][]

Input : [[][]]

Output : 0

The string is already balanced.

1. There are given n ropes of different lengths, we need to connect these ropes into one rope. The cost to connect two ropes is equal to the sum of their lengths. We need to connect the ropes with minimum cost.

For example, if we are given 4 ropes of lengths 4, 3, 2, and 6. We can connect the ropes in the following ways.   
1) First, connect ropes of lengths 2 and 3. Now we have three ropes of lengths 4, 6, and 5.   
2) Now connect ropes of lengths 4 and 5. Now we have two ropes of lengths 6 and 9.   
3) Finally connect the two ropes and all ropes have connected.  
  
Total cost for connecting all ropes is 5 + 9 + 15 = 29. This is the optimized cost for connecting ropes. Other ways of connecting ropes would always have same or more cost. For example, if we connect 4 and 6 first (we get three strings of 3, 2, and 10), then connect 10 and 3 (we get two strings of 13 and 2). Finally, we connect 13 and 2. Total cost in this way is 10 + 13 + 15 = 38.

1. Find the minimum sum of Products of two arrays of the same size, given that k modifications are allowed on the first array. In each modification, one array element of the first array can either be increased or decreased by 2.  
   Examples:

**Input :** a[] = {1, 2, -3}

b[] = {-2, 3, -5}

k = 5

**Output :** -31

**Explanation:**

Here n = 3 and k = 5.

So, we modified a[2], which is -3 and increased it by 10 (as 5 modifications are allowed).

Final sum will be:

(1 \* -2) + (2 \* 3) + (7 \* -5)

-2 + 6 - 35

-31

(which is the minimum sum of the array with given conditions)

**Input :** a[] = {2, 3, 4, 5, 4}

b[] = {3, 4, 2, 3, 2}

**Output :** 25

**Explanation:**

Here, total numbers are 5 and total modifications allowed are 3. So, modify a[1], which is 3 and decreased it by 6 (as 3 modifications are allowed).

Final sum will be :

(2 \* 3) + (-3 \* 4) + (4 \* 2) + (5 \* 3) + (4 \* 2)

6 – 12 + 8 + 15 + 8

25

(which is the minimum sum of the array with given conditions)

1. You are given an undirected tree with N nodes rooted at node 1. Every node has a value A[i] assigned to it. You are required to answer Q queries of the following type:

* UVX
  + Select a subtree with U as the root node and subtree with V as the root node and swap their positions. That is, detach both the subtrees and swap their positions.
  + If node U is an ancestor of node V or node V is an ancestor of node U, the above Swap operation is not performed.
  + Find the sum of node values present in the subtree rooted at node X.
  + If the Swap operation is performed, then redo this operation. That is, swap the subtree with U as the root node and subtree with V as the root node.

Determine the required answer for Q queries.

Note

* Assume 1-based indexing.
* A node u is said to be an ancestor of node v if node u lies on a simple path between the root and node v.
* Here, Redo means re-doing the operation performed.

**Input format**

* The first line contains a single integer T that denotes the number of test cases.
* For each test case:
  + The first line contains an integer N.
  + The second line contains N space-separated integers denoting array A.
  + The next N−1 line contains two space-separated integers uv denoting an edge between node u and v.
  + The next line contains an integer Q.
  + The next Q lines contain three space-separated integers UVX denoting the query.

**Output format**

For each test case, print Q space-separated integers denoting the sum of node values present in the subtree rooted at node X after the swap operation is performed. Print the output for each test case in a new line.

**Sample Input**

2

5

4 1 4 2 3

1 2

2 3

2 4

1 5

1

3 5 2

5

4 3 1 56 5

1 2

2 3

3 4

4 5

2

1 5 2

3 5 3

**Sample Output**

6

65 62

:

**Explanation**

**For first test case**

* For query *1: U = 3, V = 5, X = 2*
  + After swap operation, edges will be *[(1, 2), (2, 5), (2, 4), (1, 3)]*
  + Nodes in the subtree rooted at node *2* are *2, 4, 5.*
  + Sum of node values is *1 + 2 + 3 = 6.*
  + Redo the swap operation, edges will be *[(1, 2), (2, 3), (2, 4), (1, 5)].*
* Thus, the required answer is *[6].*

**For second test case**

* For query *1: U = 1, V = 5, X = 2*
  + Swap operation is not performed as node *U* is the ancestor of node *V*.
  + Nodes in the subtree rooted at node *2* are *2, 3, 4, 5.*
  + Sum of node values is *3 + 1 + 56 + 5 = 65.*
* For query 2*: U = 3, V = 5, X = 3*
  + Swap operation is not performed as node *U* is the ancestor of node *V*.
  + Nodes in the subtree rooted at node *3* are *3, 4, 5.*
  + Sum of node values is *1 + 56 + 5 = 62.*
* Thus, the required answer is *[65, 62].*